An Adaptive Hierarchical Domain Decomposition Method for Parallel Contact Dynamics Simulations of Granular Materials

Zahra Shojaaee*, M. Reza Shaebani, Lothar Brendel, János Török, Dietrich E. Wolf

Computational and Statistical Physics Group, Department of Theoretical Physics, University of Duisburg-Essen, 47048 Duisburg, Germany

Abstract

A fully parallel version of the Contact Dynamics (CD) method is presented in this paper. For large enough systems, 100% efficiency has been demonstrated for up to 256 processors using a hierarchical domain decomposition with dynamic load balancing. The iterative scheme to calculate the contact forces is left domain-wise sequential, with data exchange after each iteration step, which ensures its stability. The number of additional iterations required for convergence by the partially parallel updates at the domain boundaries becomes negligible with increasing number of particles, which allows for an effective parallelization. Compared to the sequential implementation, we found no influence of the parallelization on simulation results.

Keywords: Contact dynamics method, Granular materials, Hierarchical domain decomposition, Load balancing, MPI library
PACS: 45.70.-n, 02.70.-c, 45.10.-b

1. Introduction

Discrete element method simulations have been widely employed in scientific studies and industrial applications to understand the behavior of complex many-particle systems such as granular materials. The common property of these methods is that the time evolution of the system is treated on the level of individual particles, i.e. the trajectory of each particle is calculated by integrating its equations of motion. Among the discrete element methods, *soft particle molecular dynamics* (MD) [1, 2], *event driven* (ED) [3, 4] and *contact dynamics* (CD) [5, 6, 7, 8] are often used for simulating granular media.

Molecular dynamics is the most widely used algorithm for discrete element simulations. For granular materials, the contact forces between the soft particles stem from visco-elastic force laws. Interactions are local, therefore efficient parallelization is possible [9, 10, 11, 12] with 100% efficiency for large systems (Throughout the paper, the performance of a parallel algorithm is quantified by the usual quantities: the *speedup* $\mathcal{S}(N_p)$, which is the ratio of the run time of the non-parallel version on a single processor to the run time of the parallel version on N_p processors, and the *efficiency* $\mathcal{E}=\mathcal{S}/N_p\times100\%$). The time step and therefore the speed of MD simulations is strongly limited by the stiffness of the particles, as collisions must be sufficiently resolved in time. Molecular dynamics is efficient for dense systems of soft particles, but much less so for hard particles and dilute systems.

The *event driven dynamics* [13, 14] considers particle interactions of negligible duration compared to the time between collisions. Particle collisions are thus treated as instantaneous events, and trajectories are analytically integrated in between. This makes ED favorable in dilute granular systems, where the above condition holds. The parallelization of this algorithm poses extreme difficulties, since the collisional events are taken from a *global* list, which in turn is changed by the actual collision. In general, a naive domain

*Corresponding author
Email address: zahra.shojaaee@uni-duisburg-essen.de (Zahra Shojaaee)

decomposition leads to causality problems. The algorithm presented in [15] conserves causality by reverting to an older state when violated. The best efficiency reached so far is a speedup proportional to the square root of the number of processors [15].

In contrast to ED, lasting contacts between rigid bodies are considered in the realm of *(multi)-rigid-body dynamics*. Common to all its realizations is the treatment of contact forces as constraint forces, preventing interpenetration and, to a certain extent in the case of frictional contacts, sliding. When applying the rigid body modelling to problems like e.g. robotics [16, 17] or granular media [18, 19, 20, 21], different algorithms can in principle be used. Approximations with respect to the constraint of dry Coulomb friction enable the usage of powerful standard techniques for *linear complementary problems* (LCP) [22]. Other algorithms keep the isotropic friction cone, using a solver based on a modified time stepping scheme leading to a *cone complementary problem* (CCP) for the simulation of frictional contact dynamics [23]. Other approximations, leading to *fast frictional dynamics* (FFD) [24], yield a computational cost being only linear in the number of contacts and thus allow for impressive system sizes in terms of the number of particles. For investigations of e.g. the stress field in granular media, these approximations are prohibitive, though, and thus the *non-smooth contact dynamics* method [7], or commonly just *contact dynamics*, is widely employed. We will sketch the principle of this iterative procedure in section 2.1. Parallelization of the FFD method is straightforward and efficient [25, 26], on the other hand, the parallel version suffers also from the undesired approximations. The parallel implementation of the CCP algorithm by the use of the Graphics Processing Unit (GPU) for large-scale multibody dynamics simulations is presented in [27]. In the present work we investigate the impact of the parallelization on the numerical solution of the CD method going beyond [25, 26, 27].

Providing a parallel CD code is motivated by the need for large-scale simulations of dense granular systems of hard particles. The computation time even scales as $\mathcal{O}(N^{1+2/d})$ with the number of particles in CD [8] (d is the dimension of the system), while it grows linearly with N in MD. However, parallelization of CD poses difficulties as in general the most time consuming part of the algorithm is a global iteration procedure, which cannot be performed completely in parallel. So far, a static geometrical domain decomposition method has been proposed in Ref. [28], and a partially parallel version is introduced in Ref. [29], where only the iterative solver is distributed between shared memory CPUs. In the former work, the force calculation is studied just on 8 processors and in the latter, already with 16 processors the performance efficiency is below 70%. None of these studies deal with computational load balancing during the execution of the code.

There is a large variety of domain decomposition methods proposed for parallel particle simulations in the literature, from *Voronoi tessellation* [30] to *orthogonal recursive bisection* (ORB) [31, 32]. For the parallelization of CD the size of the interfaces between domains is more crucial than for MD, since besides communication overhead it also influences the parallel/sequential nature of the global iteration. So the ORB methods are the most suited for the CD code together with adaptive load balancing approaches [33], which is not only important in heterogeneous clusters but also in the case of changing simulation setup and local particle/contact density.

In the present work, we introduce a completely parallel version of the contact dynamics method using MPI communication with orthogonal recursive bisection domain decomposition for an arbitrary number of processors. The method minimizes the computational costs by optimizing the surface-to-volume ratio of the subdomains, and it is coupled with an adaptive load balancing procedure. The validation of our code is done by numerical simulations of different test systems. We presented our implementation in two dimensions and for spherical particles. However, our code is also capable of handling polygonal particles and the extension to three dimensions is straightforward.

This article is organized as follows. The contact dynamics method is described briefly in Sec. 2 and particular attention is paid to the numerical stability of the sequential and parallel update schemes, and to the identification of the most time consuming parts of the code. In Sec. 3 we present an adaptive domain decomposition method, and implement it in a parallel version of the CD algorithm. The results of some test simulations with respect to the performance of the parallel CD code are presented in Sec. 4 and the effect of our parallelization approach on the physical properties of the solutions are investigated. We conclude the paper in Sec. 5 with a summary of the work and a brief discussion.

2. Contact Dynamics Method

2.1. A brief description of the CD algorithm

In this section, we present the basic principles of the CD method [7] in a language closer to MD and with special emphasis on those parts where changes are applied in the parallel version of the code. For more details and a broader overview cf. [7, 34]. The central point is that the forces are not calculated from particle deformation, instead they are obtained from the constraints of impenetrability and friction laws. Imposing constraints requires implicit forces, which are calculated to counteract all movement that would cause constraint violation.

In general for molecular dynamics, where trajectories are smooth (soft particle model), simulation codes use second (or higher) order schemes to integrate the particle positions. In CD method, the non-smooth mechanics (hard particle limit) requires strong discontinuity, which can only be achieved by first order schemes. Thus we apply a *first-order Euler* scheme for the time stepping of particle i:

$$\vec{v}_i(t+\Delta t) = \vec{v}_i(t) + \frac{1}{m_i}\vec{F}_i\,\Delta t, \tag{1}$$

$$\vec{r}_i(t+\Delta t) = \vec{r}_i(t) + \vec{v}_i(t+\Delta t)\Delta t\,, \tag{2}$$

which determines the new velocity $\vec{v}_i$ and position $\vec{r}_i$ of the center of mass of the particle after a time step Δt. The effective force on particle i is denoted by F_i. The scheme is semi-implicit in the sense that the right-hand-side velocities are (necessarily) the ones at time $t+\Delta t$ while forces other than the constraint forces may be treated implicitly or explicitly. The size of the time step Δt is chosen such that the relative displacement of the neighboring particles during one time step is much smaller compared to the size of particles and to the radius of curvature of contacting surfaces. Similar equations are used for the rotational degrees of freedom, i.e. to obtain the new angular velocity $\vec{\omega}_i(t+\Delta t)$ (caused by the new torque $\vec{T}_i(t+\Delta t)$), and the new orientation of particle i.

For simplicity, in the following we assume that particles are dry and non-cohesive having only unilateral repulsive contact forces. Furthermore, we assume perfectly inelastic particles, which remain in contact after collision and do not rebounce. The implicit scheme must fulfill the following two constraints:

(i) the *impenetrability* condition: the overlapping of two adjacent particles has to be prevented by the contact force between them.

(ii) the *no-slip* condition: the contact force has to keep the contact from sliding below the Coulomb friction limit, i.e. the tangential component of the contact force cannot be larger than the friction coefficient times the normal force.

The contact forces should be calculated in such a way that the constraint conditions are satisfied at time $t+\Delta t$, for the new particle configuration [8]. Once the total force and torque acting on the particles, including the external forces and also the contact forces from the adjacent particles, are determined, one can let the system evolve from time t to $t+\Delta t$.

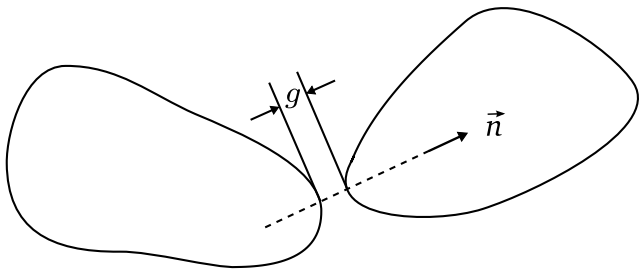

Figure 1: Schematic picture showing two adjacent rigid particles.

Let us now consider a pair of neighboring rigid particles in contact or with a small gap between them as shown in Fig. 1. We define $\vec{n}$ as the unit vector along the shortest path of length g between the surfaces of the two particles. The relative velocity of the closest points is called the *relative velocity of the contact* $\vec{v}_g$. In the case that the particles are in contact, the gap g equals to zero, and $\vec{n}$ denotes the contact normal.

We first assume that there will be no interaction between the two particles at $t+\Delta t$, i.e. the new contact force $\vec{R}(t+\Delta t)$ equals to zero. This allows the calculation of a hypothetical new relative velocity of the two particles $\vec{v}_{g,0}(t+\Delta t)$ through Eq. (1), only affected by the remaining forces on the two particles. The new gap reads as:

$$g(t+\Delta t) = g(t) + \vec{v}_{g,0}(t+\Delta t)\cdot\vec{n}\Delta t. \tag{3}$$

If the new gap stays indeed positive ($0<g(t+\Delta t)$) then no contact is formed and the zero contact force is kept: $\vec{R}(t+\Delta t)=0$.

On the other hand, if the gap turns out to be negative ($g(t+\Delta t) \leq 0$), a finite contact force must be applied. First, we determine the new relative velocity from the condition that the particles remain in contact after the collision,

$$0 \equiv g(t+\Delta t)\vec{n} = g(t)\vec{n} + \vec{v}_g(t+\Delta t)\Delta t \tag{4}$$

Here we assume sticking contacts with no relative velocity in the tangential direction ($\vec{v}_g^{\,t}(t+\Delta t)=0$), which implies that the Coulomb condition holds. The new contact force satisfying the impenetrability can be obtained using Eq. (1) as

$$\vec{R}(t+\Delta t) = \frac{\mathbf{M}}{\Delta t}\Big(\vec{v}_g(t+\Delta t) - \vec{v}_{g,0}(t+\Delta t)\Big) = \frac{-\mathbf{M}}{\Delta t}\left(\frac{g(t)}{\Delta t}\vec{n} + \vec{v}_{g,0}(t+\Delta t)\right) \tag{5}$$

where the mass matrix $\mathbf{M}$, which is built up from the masses and moments of inertia of both particles [8], reflects the inertia of the particle pair in the sense that $\mathbf{M}^{-1}\vec{R}$ corresponds to the relative acceleration of the contacting surfaces induced by the contact force $\vec{R}$.

At this point, we have to check for the second constraint: the Coulomb friction. Let us first define the normal and tangential contact forces:

$$\begin{aligned} R_n(t) &\equiv \vec{R}(t)\cdot\vec{n}\ , \\ \vec{R}_t(t) &\equiv \vec{R}(t) - R_n(t)\vec{n}\ . \end{aligned} \tag{6}$$

Then the Coulomb inequality reads as

$$\Big|\vec{R}_t(t+\Delta t)\Big| \leq \mu R_n(t+\Delta t)\,, \tag{7}$$

where μ is the friction coefficient (being the same for static and dynamic friction, the standard Coulomb model of dry friction [34]). If the inequality (7) holds true, then we have already got the correct contact forces. Otherwise, the contact is sliding, i.e. $\vec{v}_g(t+\Delta t)$ has a tangential component and Eq. (4) reads

$$0 \equiv g(t+\Delta t) = g(t) + \vec{n}\cdot\vec{v}_g(t+\Delta t)\Delta t\,, \tag{8}$$

which determines the normal component of $\vec{v}_g(t+\Delta t)$. The remaining five unknowns, three components of the contact force $\vec{R}(t+\Delta t)$ and two tangential components of the relative velocity, are determined by the following two equations:

(i) Impenetrability by combining Eqs. (4) and (5)

$$\vec{R}(t+\Delta t)=\frac{\mathbf{M}}{\Delta t}\left(-\frac{g(t)}{\Delta t}\vec{n} + \vec{v}_g^{\,t}(t+\Delta t) - \vec{v}_{g,0}(t+\Delta t)\right). \tag{9}$$

(ii) Coulomb condition

$$\vec{R}_t(t+\Delta t) = -\mu R_n(t+\Delta t)\frac{\vec{v}_g^{\,t}(t+\Delta t)}{\left|\vec{v}_g^{\,t}(t+\Delta t)\right|}. \tag{10}$$

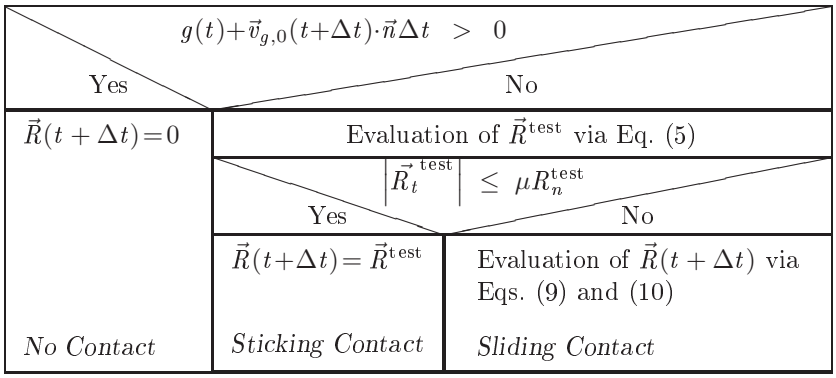

Figure 2: The force calculation process for a single contact.

In two dimensions and for spheres in three dimensions, these equations have an explicit analytical solution, otherwise one has to resort to a numerical one[7].

Figure 2 summarizes the force calculation process for a single incipient or existing contact. Assuming that all other forces acting on the participating particles are known, the Nassi-Shneiderman diagram [35] in Fig. 2 enables us to determine the contact force.

The above process assumes that apart from the contact forces all other interactions are known for the selected two particles. However, in dense granular media, many particles interact simultaneously and form a contact network, which may even span the whole system. In such cases, the contact forces cannot be determined locally because each unknown contact force depends on the adjacent unknown contact forces acting on the particles. In order to find the unilateral frictional forces throughout the entire contact network, an *iterative* method is mostly used in CD as follows: At each iteration step, we choose the contacts randomly one by one and calculate the new contact force considering the surrounding contact forces to be already the correct ones. It is natural to update the contact forces sequentially in the sense that each freshly calculated force is immediately used for further force calculations. One iteration step does not provide a globally consistent solution, but slightly approaches it. Therefore, the iteration has to be repeated many times until the forces relax towards an admissible state. To assess whether or not the convergence is achieved, we measure the relative change of each contact force $\vec{R}_i$ at each iteration step j, as well as the relative change in the average contact force $\vec{R}_{\rm avg}$ at this iteration step. Generally, we choose one of the following convergence criteria to stop the force calculation procedure:

(I) *local convergence test*: if, at least for 90% of the contacts, the following condition holds

$$\frac{(\vec{R}_i^{\,j}-\vec{R}_i^{\,j-1})^2}{(\vec{R}_i^{\,j}+\vec{R}_i^{\,j-1})^2} < \alpha,$$

and the rest of contacts fulfill

$$(\vec{R}_i^{\,j}-\vec{R}_i^{\,j-1})^2 < \alpha(\vec{R}_{\rm avg}^{\,j-1})^2.$$

(II) *global convergence test*: if the relative change in the average contact force falls below the threshold value α, i.e.

$$\frac{(\vec{R}_{\rm avg}^{\,j}-\vec{R}_{\rm avg}^{\,j-1})^2}{(\vec{R}_{\rm avg}^{\,j}+\vec{R}_{\rm avg}^{\,j-1})^2} < \alpha.$$

We have chosen α=10^{-6} in all simulations.

The precision of the solution increases smoothly with the number of iterations N_I, with the exact solution being only reached for $N_I \to \infty$. Of course we stop at finite N_I. It is optional to use a fixed number of

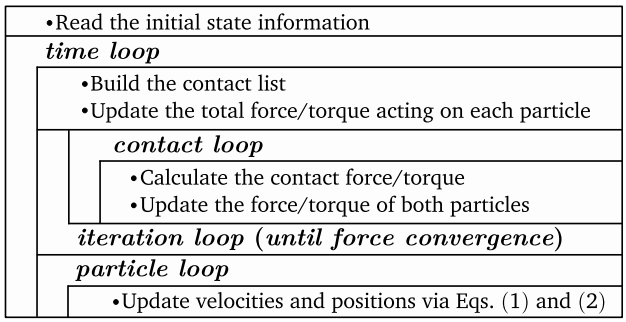

Figure 3: The diagram of the main steps of the contact dynamics algorithm.

iterations at each time step, or to prescribe a given precision to the contact force convergence and let N_I to vary at each time step.

Breaking the iteration loop after finite iteration steps is an inevitable source of numerical error in contact dynamics simulations, which mainly results in overlap of the particles and in spurious elastic behavior [36]. Occurring oscillations are a sign that the iterations were not run long enough to allow the force information appearing on one side of the system to reach the other side. This effect should be avoided and the number of iterations should be chosen correspondingly [36].

Once the iteration is stopped, one has to update the particle positions based on the freshly calculated forces acting on the particles. Figure 3 concludes this section with a diagram depicting the basic steps of the contact dynamics algorithm.

The question of successful convergence in general is difficult (cf. [37, 38]) but in practice convergence turns out to be given and hence the CD method has been experimentally validated in different instances, e.g. as in calculating the normal contact force distribution in static 2D and 3D granular packings especially for weak forces that are experimentally difficult to access [18], investigating the dynamics of granular flows e.g. monitoring the evolution of the contact orientations and shear band formation in a biaxial shear cell [19], studying the mechanical properties of cohesive powders [20], and predicting the refraction of shear zone in layered granular media [21].

2.2. CPU time analysis

The CD algorithm described in the previous section has three main parts: (i) The contact detection, (ii) the force calculation (iteration), (iii) the time evolution. In this section we analyze the CPU consumption of all these parts.

Given a system and the contact detection algorithm, the time consumption of parts (i) and (iii) can be easily estimated. On the other hand, the computational resource needed by part (ii) is strongly influenced by the number of iterations. If one uses extremely high values of N_I, part (ii) will dominate the CPU usage. This led Renouf *et al.* [29] to the conclusion that parallelizing the force calculation is enough.

Our view is that the situation is more delicate and it is demonstrated by a simulation in which diluted granular material is compressed until a dense packing is reached [39]. The system consists of 1000 polydisperse disks in two dimensions with friction coefficient μ=0.5. The stopping criteria for the iteration was the fulfillment of any of the two conditions:

(1) The global convergence criterion is fulfilled (see Sec. 2.1 for details).
(2) $N_I \geq 200$

Figure 4 shows the evolution of the relative CPU time consumption of the different parts of the algorithm. The time stepping contribution always remains less than 5%, and the rest is divided between the other two subroutines. Initially, the contact detection task consumes the majority of the computational time. After a while, clusters of contacting particles form, and the cost of force calculation increases and the iterative

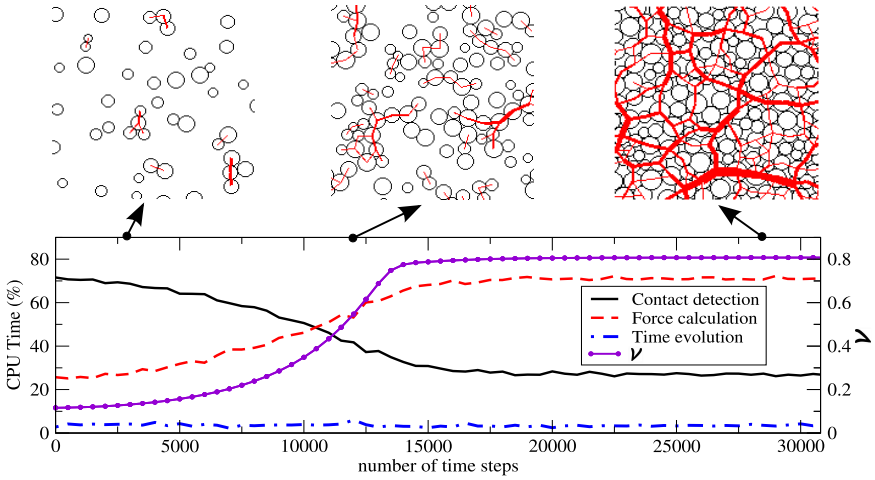

Figure 4: (color online) The percentage of CPU time consumption (lines) and the packing fraction ν (purple line, full circles) as a function of time. The insets show typical configurations of particles at different packing fractions. The thickness of the inter-center connecting red lines is proportional to the magnitude of the contact force.

solver gradually becomes the most time consuming part of the code. Note that the contribution of the solver saturates to 70% of the total elapsed time. If only the force calculation part is executed in parallel, even with $\mathcal{E}_{\text{force}} = 100\%$, the remaining 30% non-parallel portions set an upper limit to the overall efficiency $\mathcal{E}$ and the speedup $\mathcal{S}$ of the code ($\mathcal{E}_{\max} \approx 80\%$ and $\mathcal{S}_{\max} \approx 4$). Therefore, we aim to provide a fully parallel version of CD which operates efficiently in all density regimes.

2.3. Sequential versus parallel update scheme

As we pointed out in Sec. 2.1, the problem of finding the unilateral frictional contact forces that satisfy the constraint conditions cannot be solved locally in a dense granular system. In order to evaluate the new value of a single contact, one has to know the new values of the adjacent contact forces, which are unknown as well, i.e. all contact forces are coupled in a cluster of contacting particles. Note that this is a consequence of the infinite stiffness of the particles; a single collision influences the entire network of contact forces between perfectly rigid particles. This problem is solved by iterating through all contacts many times until a given precision is reached.

Similarly to solving the Laplace equation, the information about a disturbance (e.g. collision of a new particle) appearing on one side of a cluster must diffuse through the whole cluster to satisfy the constraints. Actually, the iteration scheme is very similar to two traditional schemes for solving a set of linear equations [40], albeit with nonlinearities introduced by the change of contact states (repulsive vs. force-less, sticking vs. sliding): the Jacobi scheme and the Gauss-Seidel scheme, corresponding to parallel and sequential contact updating, respectively.

Here we denote (i) *sequential*, where the contacts are solved one by one using always the newest information available, which is a mixture of new and old values, (ii) *parallel*, where all contacts are updated using the old values, and substituted with the new ones at the end of the iteration step. Needless to say that the second case is favored for parallel applications but instabilities may appear (like when combining the Jacobi scheme with Successive Over-Relaxation [40]). To study its impact, we investigated a mixed method, where a fraction p of the contacts are updated in parallel and the rest sequentially. First, a static homogeneous packing is generated by applying an external confining pressure [39]. Next, the inter-particle forces are set to zero, while the positions of the particles and the boundary conditions are kept fixed. Now the code recalculates the contact forces within one time step with an unconstrained number of iterations until the convergence is reached. We check how many iteration steps are needed to find a consistent equilibrium solution with a given accuracy threshold. The results are shown in Fig. 5(a).

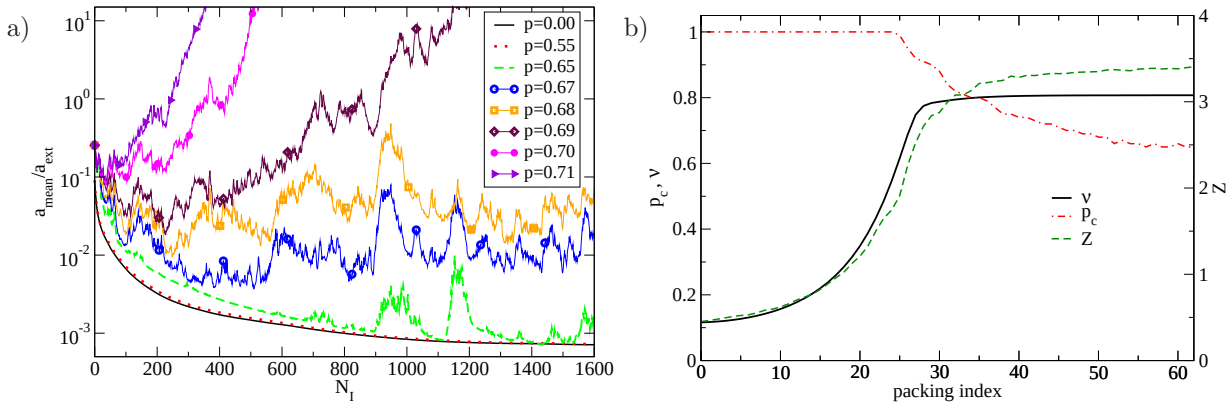

Figure 5: (color online) (a) The mean acceleration of the particles a_{mean} scaled by $a_{\text{ext}}=2\bar{r}P_{\text{ext}}/\bar{m}$ (where $\bar{r}$ and $\bar{m}$ are the mean particle radius and mass, respectively, and P_{ext} is the external pressure) in terms of the number of iterations N_I for several values of the "parallelness" p (cf. text). These results belong to the dense packing in the right panel of Fig. 4. (b) The critical parallelness ratio p_c, the average coordination number Z, and the packing fraction ν for several configurations obtained during the time evolution of the system in the simulation presented in Fig. 4.

It turns out that, on average, the number of iterations N_I to reach a given accuracy level increases with increasing p. For high values of p, fluctuations appear and beyond $p_c \approx 0.65$ the iterative solver is practically unable to find a consistent solution. We discuss the consequence of this behavior for the parallel version of CD in Secs. 3 and 4.

In order to investigate the dependence of p_c on the properties of the contact network, we take snapshots of the structure during the time evolution of the system in the simulation presented in Fig. 4. The same procedure as mentioned above is then applied to each of these samples to obtain p_c. The results are shown in Fig. 5(b). In dilute systems, the contacts form small isolated islands and the resulting set of equations is decomposed into smaller independent sets, so that even a completely parallel update scheme (p_c=1.0) can be tolerated. However, the contact network in dense systems forms a set of fully coupled nonlinear equations which converges only if the parallelness factor p is less than $p_c{\sim}0.65$. By varying the system size and the friction coefficient, we conclude that p_c is mainly influenced by the degree of coupling between the equations which is reflected in the connectivity of the sample Z [see Fig. 5(b)].

Thus, the results of our numerical simulations reveal that the sequential update scheme is quite robust and the force convergence is reached smoothly, while the fully parallel update scheme is highly unstable in dense systems. However, there is a limit of parallel update for which the iteration remains stable. This is important because the domain decomposition method allows for a sequential update only in the bulk of each domain, while the boundary contacts are updated in a parallel way (cf. section 3.1). This analysis suggests that the ratio of bulk contacts to boundary ones after the decomposition should never fall below 1. Fortunately, this is assured in a domain decomposition context anyway.

3. A parallel version of the CD algorithm

3.1. The parallel algorithm

A parallel version of the CD algorithm based on the decomposition of the simulation domain is introduced in this section. The main challenge is to properly evaluate the inter-particle forces when the contact network is broken into several subnetworks assigned to different processors. The parallelization presented in this section is valid only for spherical particles (disks in 2D), but it is straightforward to extend it for other shape types.

At the beginning of the simulation, a domain decomposition function is called to divide the system between N_p processors. Regarding the fact that neither the performance of the computing environment nor

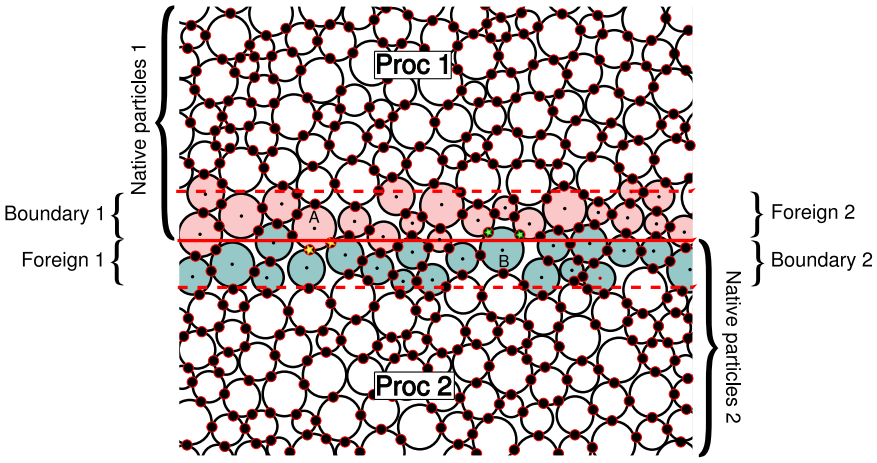

Figure 6: (color online) Schematic picture showing two neighboring processors at their common interface. Their respective domain and boundary regions are marked. Particle A is a *native* particle of processor 1 and is in contact (asterisks) with two *foreign* particles, namely *boundary* particles of processor 2. The contacts are *boundary* contacts of processor 2 and thus *foreign* ones to processor 1. Particle B is a *boundary* particle of processor 2 and has two contacts (asterisks) located inside the domain of processor 1, i.e. they belong to the latter's *boundary* contacts.

the density distribution and the internal dynamics of the system are known initially, a uniform distribution for all relevant factors is assumed and initially the simulation domain is geometrically divided into N_p parts with the same volume. The details of the hierarchical decomposition method are explained in Sec. 3.2.

After establishing the domains, the particles are distributed among the processors. Each processor maintains its set of *native* particles, the center of mass of which lie within its domain. The next task is to identify in each domain the *boundary* particles, i.e. those particles which may be in contact with particles in other domains, as this information should be passed to the neighbors. Two particles may come into contact if the gap is smaller than $2v_{\max}\Delta t$, where $v_{\max}$ is the maximum velocity in the whole system. So the maximal distance between the centers of mass of two particles, which may come into contact is

$$d \leq 2r_{\max} + 2v_{\max}\Delta t, \tag{11}$$

where $r_{\max}$ is the radius of the largest particles. This distance also defines the width of the *boundary region* in which particles may have contact with particles outside a processor's domain, see also Fig. 6.

While $r_{\max}$ is constant during the simulation, $v_{\max}$ varies in time and space. For reasons described in Sec. 3.2, we use a global upper limit ℓ for the boundary size, which is unchanged during the whole simulation. It was explained in Sec. 2.1, that the displacement of the particles must be small compared to particle size for contact dynamics to be valid. Therefore it is legitimate to define the upper limit for the particle displacement to be $0.1r_{\max}$ and thus use the boundary size

$$\ell = 2.2r_{\max}\,. \tag{12}$$

Hence, a small amount of in principle irrelevant neighboring information is transferred. This is dominated by other effects, though, as will be shown in Sec. 3.2.

After the identification of the *boundary* particles, their relevant data is sent to the corresponding neighbor processors, which keep the information of these (to them) *foreign* particles. Since sender and receiver will always agree about the forces acting on these particles, the receiver can evolve their state on its own.

The next step is to identify actual and possible contacts between both *native* and *foreign* particles. A position is assigned to each contact, which is the middle of the gap (see Fig. 1). Obviously, for particles in touch, this is the contact point. Each processor builds a list of *native* contacts for the iteration loop

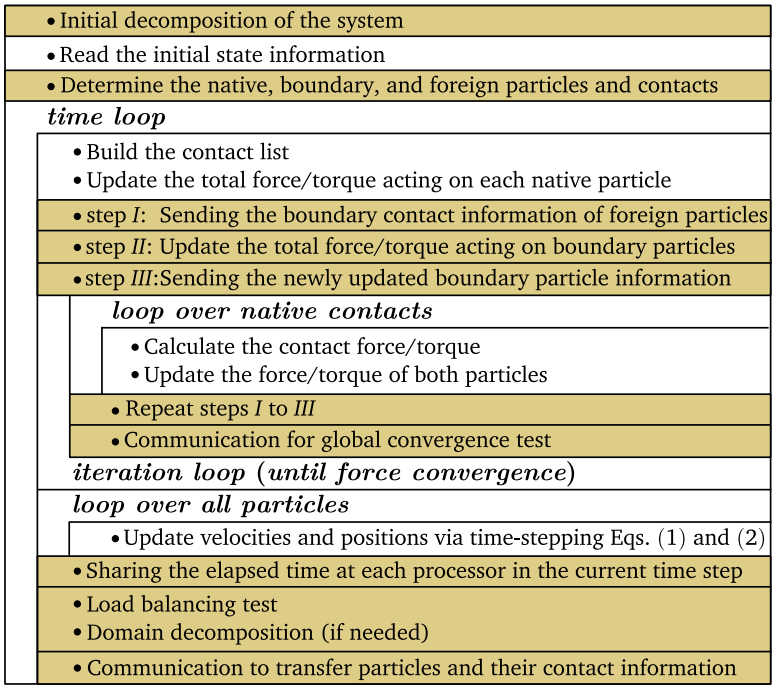

Figure 7: (color online) The diagram of the parallel version of CD. The colored regions correspond to the new parts compared to the original CD algorithm shown in Fig. 3.

exclusively from contacts lying in its domain. The remaining ones are called *foreign* contacts and are in turn *boundary* contacts of neighboring processors. During an iteration sweep, they will not be updated but their forces enter the force calculation algorithm. Only at the end of the sweep, each processor sends the new forces of its *boundary* contacts to its corresponding neighbor. This means that during an iteration sweep, foreign contacts always have the values from the last iteration, while native contacts are gradually updated realizing a mixture of parallel and sequential update.

The convergence of the force calculation has to be checked after each iteration sweep. This should be a global test, since the convergence in different subdomains may be achieved at different iteration steps. This task can only be completed by a single processor. Therefore, the necessary data is collected and submitted to the root processor, which makes a decision whether the iteration should continue or the convergence is good enough and time stepping can take place. If further iterations are necessary, then only boundary *contact* information are exchanged among neighbors, as particles do not move within the iteration loop. With new foreign contact values, each processor can perform the next iteration sweep. If the iteration loop has finished, the particles are displaced according to the implicit Euler scheme of Eqs. (1) and (2). Every processor is responsible for its own native particles (but evolves its copies of foreign particles as well).

Before starting the next time step, we have to take care of the load balancing: Every processor broadcasts its own elapsed CPU time, which provides the required information to run the load balancing function. The detailed description of this function is presented in Sec. 3.3. If the load balancing function redivides the simulation box, then each processor has to compare its own particle positions to the new domain coordinates of all other processors to determine to which processor each particle has to be sent. This re-association of particles takes place also without domain redivision as particles change domains simply due to their dynamics.

Figure 7 summarizes the parallel algorithm. The main differences (highlighted in the diagram) are that (i) at certain points data must be sent or received to neighboring domains; (ii) the iteration scheme updates only native contacts gradually, while foreign contacts are refreshed only after a complete iteration sweep; (iii) load balancing and domain redivision checks take place at the end of the time step.

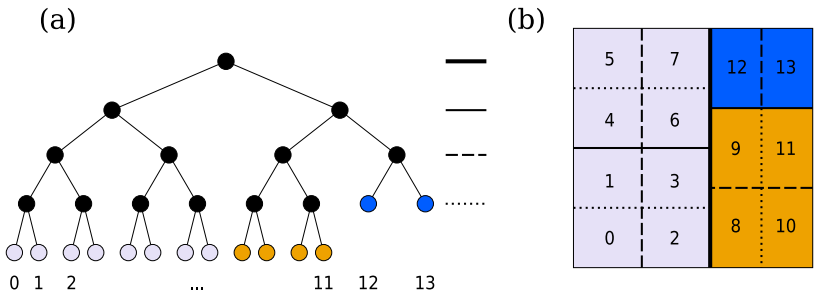

Figure 8: (color online) An initial hierarchical decomposition of the simulation domain for $N_p = 14$.

A mixture of the sequential and the parallel update scheme occurs for a fraction of the contacts. This fraction depends on the surface-to-volume ratio of the subdomain. As discussed in Sec. 2.3, a mixed update can become unstable if the contribution of the parallel update exceeds a threshold of order unity. This limitation coincides with the standard limitation of parallel computation that the boundary region should be negligible compared to the bulk. In this sense, for reasonably large systems, we do not expect any instability impact due to the parallel update. Nevertheless, this issue is investigated in Sec. 4.3.

In the next section we introduce a hierarchical domain decomposition method, which finds the best way to arrange the orientation and location of the interfaces so that the surface-to-volume ratio is minimal for a given number of processors.

3.2. Hierarchical domain decomposition

Before describing the domain decomposition, we have to investigate the contact detection. This process, for which the brute force algorithm scales as $\mathcal{O}(N^2)$ with the number of particles, can be realized for different levels of polydispersity [41, 42, 43] within $\mathcal{O}(N)$ CPU cycles. We chose to implement the most widespread one, the cell method [41], which works well for moderate polydispersity and which is the most suitable for parallel implementation.

The cell method puts a rectangular grid of mesh size $a_x \times a_y$ on the simulation space. Each particle is assigned to its cell according to its position, and the mesh size is chosen such that the particles can only have a contact with particles from neighboring cells and their own. That means, the cell diameter has essentially the same meaning as the width of the boundary region ℓ and thus they should coincide. On the other hand, the values a_x and a_y have to be chosen such that in each direction every domain has an integer number of cells. But this would mean, in general, a different mesh size for all subdomains, which may be far from the optimal value. Therefore, it is advantageous (for large systems and moderate polydispersities) to choose a global a_x and a_y instead, and restrict the domain boundaries to this grid.

The domain decomposition method proposed in this paper is based on the *orthogonal recursive bisection* algorithm [31] with axis-aligned domain boundaries. The basis of the algorithm is the hierarchical subdivision of the system. Each division represents recursive halving of domains into two subsequent domains. The advantage of such a division is an easy implementation of load balancing, which can be realized at any level, simply by shifting one boundary.

First, we have to group the N_p processors (where N_p is not required to be a power of two) hierarchically into pairs. The division algorithm we use is the following: We start at level 0 with one *node*[1], which initially is a *leaf* (a node with no children) as well. A new level l is created by branching each node of level $l-1$ in succession into two nodes of level l, creating 2^l leaves. This continues until $2^l < N_p \leq 2^{l+1}$. Then, only $N_p - 2^l$ leaves from level l are branched from left to right, cf. Fig. 8(a).

Next, we have to assign a domain to each leaf/processor. In the beginning, having no information about the system, all domains should have the same size. Actually, their sizes equal only approximatively due to grid restriction described above, cf. Fig. 9(a). To achieve this, the recursive division of the sample is done

[1]These are abstract nodes in a tree rather than (compounds of) CPUs.

according to the tree just described. Each non-leaf node represents a bisection with areas corresponding to the number of leaves of its branches (subtrees). The direction of the cut is always chosen as to minimize the boundary length.

The hierarchical decomposition method provides the possibility of quick searches through the binary tree structure. For example, the task to find the corresponding subdomain of each particle after load balancing requires a search of order $\mathcal{O}(\log(N_p))$ for N_p processors. With respect to bookkeeping overhead, a further advantage of this decomposition scheme is that local load imbalance does not necessarily affects higher level subdomain boundaries. For example, if particle exchange takes place across a low level domain boundary only this boundary will move leaving the others untouched.

3.3. Adaptive load balancing

For homogeneous quasi-static systems, the initially equal-sized subdomains provide already a reasonably uniform load distribution, but for any other case the domain boundaries should dynamically move during the simulation. In the load balancing function, we take advantage of the possibility provided by MPI to measure the wall clock time accurately. For every time step, the processors measure the computational time spent on calculations and broadcast it, so that all processors can decide simultaneously whether or not the load balancing procedure has to be executed. To quantify the global load imbalance, the relative standard deviation of the elapsed CPU time in this time step [2] is calculated via the dimensionless quantity

$$\sigma_T \equiv \frac{1}{\langle T \rangle} \sqrt{\langle T^2 \rangle - \langle T \rangle^2}, \tag{13}$$

where the average is taken over the processors.

A threshold value σ_T^* is defined to control the function of the load balancing algorithm: If $\sigma_T < \sigma_T^*$, then the simulation is continued with the same domain configuration, otherwise load balancing must take place. This load balancing test is performed by all processors simultaneously, since all of them have the necessary data. The result being the same on all processors, no more communication is needed.

If the above test indicates load imbalance, we have to move the domain boundaries. This may happen at any non-leaf node of the domain hierarchy tree. The relevant parameter for the domain division is the *calculating capacity* of the branches, which is defined as

$$\nu_j = \sum_i \frac{V_i}{T_i}, \tag{14}$$

where T_i and V_i are the CPU time and volume of domain i, respectively, and the summation includes all *leaves* under branch j. Let us denote the two branches of a node as j and k, then the domain must be bisectioned according to

$$\tilde{\nu}_j \equiv \frac{\nu_j}{\nu_j + \nu_k} \quad \text{and} \quad \tilde{\nu}_k \equiv 1 - \tilde{\nu}_j \quad . \tag{15}$$

The above procedure is repeated for all parent nodes. If the size of a domain was changed, then all subdomain walls must be recalculated as even with perfect local load balance the orientation of the domain boundary may be subject to change. Note that boundaries must be aligned to the grid boundaries as explained in Sec. 3.2.

As an example, let us consider the situation of Fig. 8 at the node of level 0 with branch 1 to the left and branch 2 to the right. If all T_i would be the same, then $\tilde{\nu}_1 = 8/14$ and $\tilde{\nu}_2 = 6/14$, just as the initial configuration. Let us now assume that the processors 12 and 13 [top right in Fig. 8(b)] are only half as fast as the others, thus, the elapsed time is twice as much. In this case $\tilde{\nu}_1 = 8/13$ and $\tilde{\nu}_2 = 5/13$, so the thick, solid division line moves to the right. Furthermore, the thin, solid division line on the right moves up from the position 4/6 to 4/5.

[2] Assuming exclusive access to the computing resources on every processor, we identify wall clock time and CPU time throughout this work.

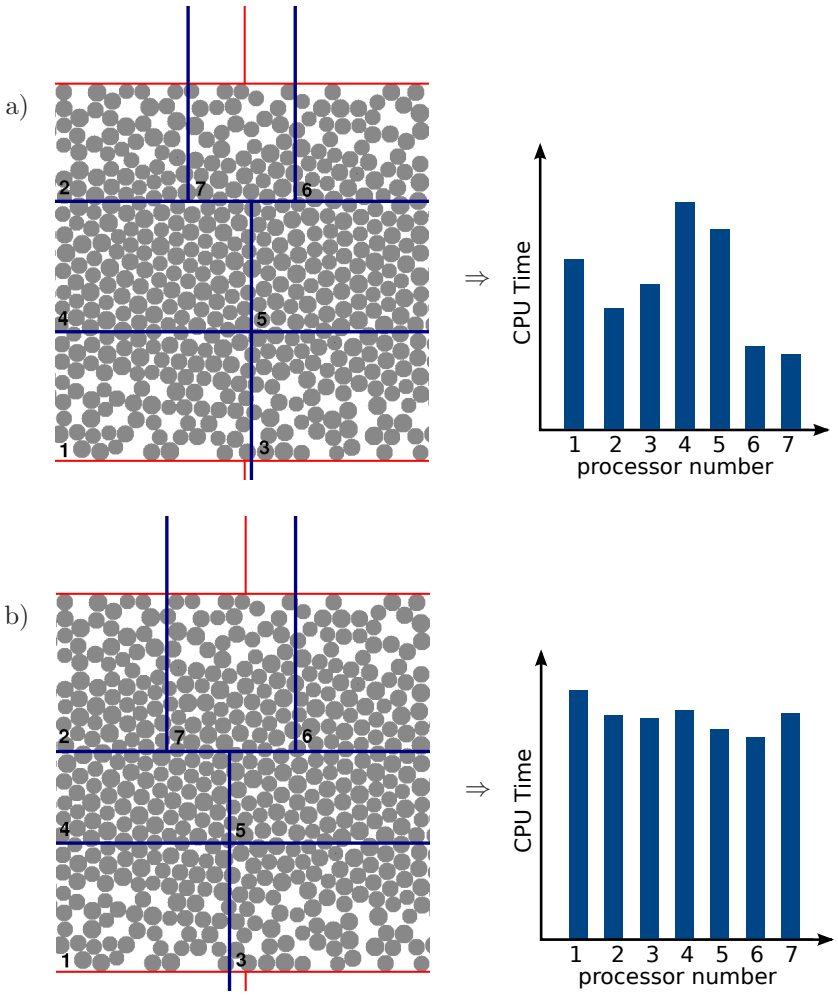

Figure 9: (color online) (a) Geometrical domain decomposition at the beginning of the simulation leads to an unbalanced distribution of the load over the processors. (b) After load balancing, the volume of the subdomains belonging to different processors vary according to the CPU time it needed in the previous time step and the load distribution over the processors becomes more even.

Figure 9 shows how load balancing improves the CPU time distribution over seven processors. The initial geometrical decomposition leads to an uneven workload distribution because of the inhomogeneous density of the original particle configuration [Fig. 9(a)]. However, the load balancing function manages to approximately equalize the CPU times in the next time step by moving the borders [Fig. 9(b)].

4. Numerical results

In the following, we present the results of test simulations for different systems performed by the parallel code. The main question to answer is how efficient is the parallel code, i.e. how much could we speed up the calculations by means of parallelization. The sensitivity of the performance to the load balancing threshold is also studied. The partially parallel updates at the domain boundaries is the main consequence of parallelization, which may make a difference in the results compared to the sequential implementation. Therefore, we investigate the impact of parallelization on the number of iterations and on the physical properties of the solutions.

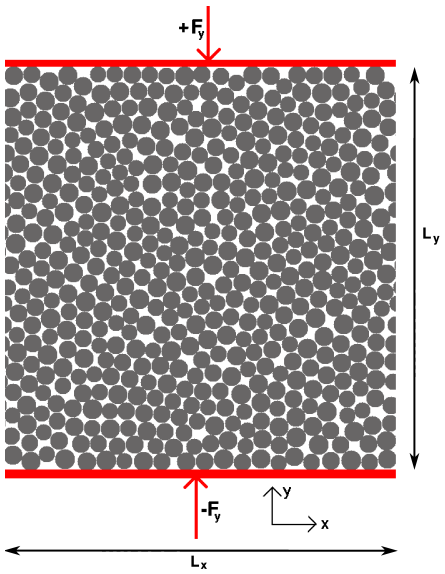

Figure 10: (color online) The simulation setup used for performance tests. The system is confined by two lateral walls in y direction (exerting a pressure of 0.25 natural units), and periodic boundary conditions are applied in x direction. The packings contain 500, 8000, and 10^6 particles with L_x=20, 20, 100 and L_y=20, 320, 10000, respectively. The polydispersity in the small and medium systems amounts to 20%, while the large system is monodisperse.

4.1. Performance of the force calculation

In this section, we test the efficiency of the parallel algorithm solely with respect to the force calculation. In general, it is the most time consuming part of the contact dynamics simulation (see Sec. 2.2), so the efficient parallelization of the iteration scheme is necessary for the overall performance.

To focus just on the force calculation, we chose test systems where large scale inhomogeneities are absent and adaptive load balancing is unnecessary. Thus, dense static packings of 500, 8000, and 10^6 particles with periodic boundary conditions in one direction and confining walls in the other were set up [see Fig. 10]. The calculations started with no information about the contact forces and the simulation was stopped when the local convergence criterion is fulfilled (see Sec. 2.1). Of course, this requires a different number of iterations depending on the system size and number of processors. In order to get rid of perturbing factors like input/output performance, we measured solely the CPU time spent in the iteration loop. Figure 11 summarizes the test results, which show that if the system is large compared to the boundary regions, the efficiency is about 100%, which is equivalent to a *linear* speedup. The smallest system is inapt for parallelization, as already for only 4 processors the boundary regions take up 20% of the particles, which induces a large communication overhead. The same fraction of boundary particles is reached around N_p=32 for the medium sized system with 8000 particles. Therefore, one would expect the same performance for N_p=4 and 32 for the small and medium sizes, respectively. In addition to the above mentioned effect, the efficiency of the medium system breaks down at N_p=24 due to special architecture of the distributed memory cluster used for simulations (Cray-XT6m with 24 cores per board), since the speed of the inter-board communications is much slower than the intra-board one. The observed efficiency values over 100% are possible through caching, which was already observed in molecular dynamics [12]. The largest system has a large computation task compared to the boundary communication, which is manifested in almost 100% efficiency. On the other hand, it is also too large for significant caching effects producing over 100% efficiency. However, a gradual increase in the efficiency is observed as the domain size (per processor) decreases with increasing the number of processors.

For the medium sized system, we also measured the overall performance including time stepping and load balancing. For this purpose, the top wall was removed and the bottom wall was pushed upwards in order to generate internal dynamical processes, which unbalances the load distribution. As shown in Fig. 11,

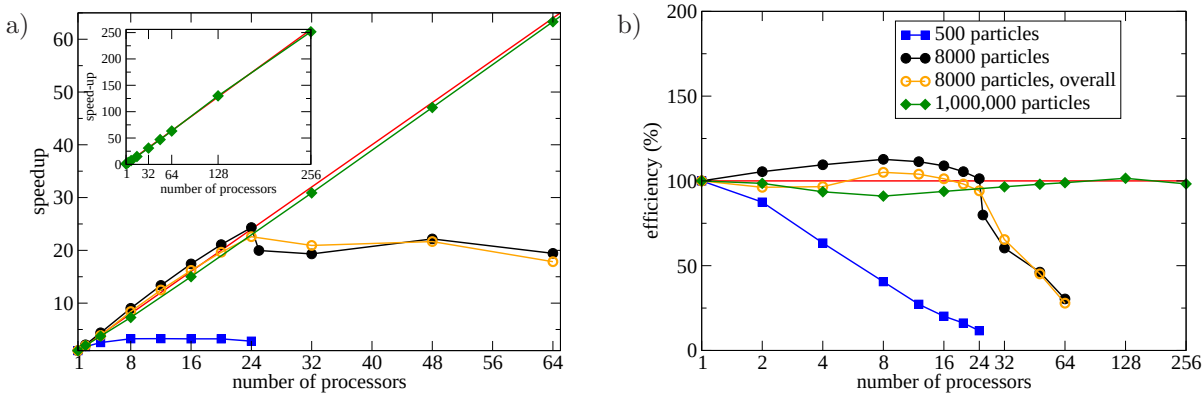

Figure 11: (color online) (a) Speedup and (b) efficiency of the force calculations for a small system with 500 particles (full squares), a medium system with 8000 particles (full circles), and a large system with 10^6 particles (full diamonds). The open circles present the overall efficiency for the medium sized system.

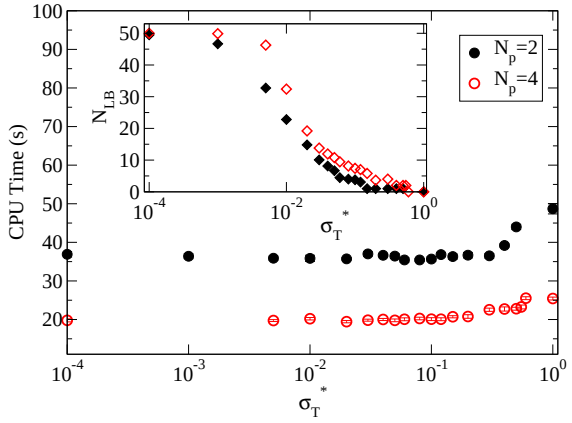

Figure 12: (color online) CPU time as a function of the load balancing threshold σ_T^*. The simulation runs over 50 time steps with 2 or 4 processors. The inset shows the number of load balancing events versus σ_T^*.

there is no significant difference in efficiency due to the fact that time stepping and contact detection are perfectly parallelizable processes.

4.2. Load balancing threshold

In Sec. 3.3, we defined the load balancing threshold σ_T^* for the relative standard deviation of the elapsed CPU time on different processors, above which load balancing takes place. While the load balancing test is performed at each time step, the frequency of load redistribution is determined by the choice of σ_T^*. On the one hand, if the subdomain redivision happens frequently, a waste of CPU time is avoided because of even load distribution. On the other hand, the change of domain boundaries requires extra communication and administration. Doing this too often leads to unwanted overhead.

For load balancing, contact dynamics has the advantage, compared to other DEM methods, that the configuration changes rather infrequently (with respect to CPU time), because the force calculation with typically 50−200 iteration sweeps (for reasonably accurate precision of contact forces) dominates the computation. Thus, even taking the minimal value of σ_T^*=0 does not lead to measurable overhead. Moreover, in our implementation the domain boundaries must be on the cell grid, which avoids unnecessary small displacements of the domain walls. Hence, the optimal value of σ_T^* is the minimal one as shown in Fig. 12.

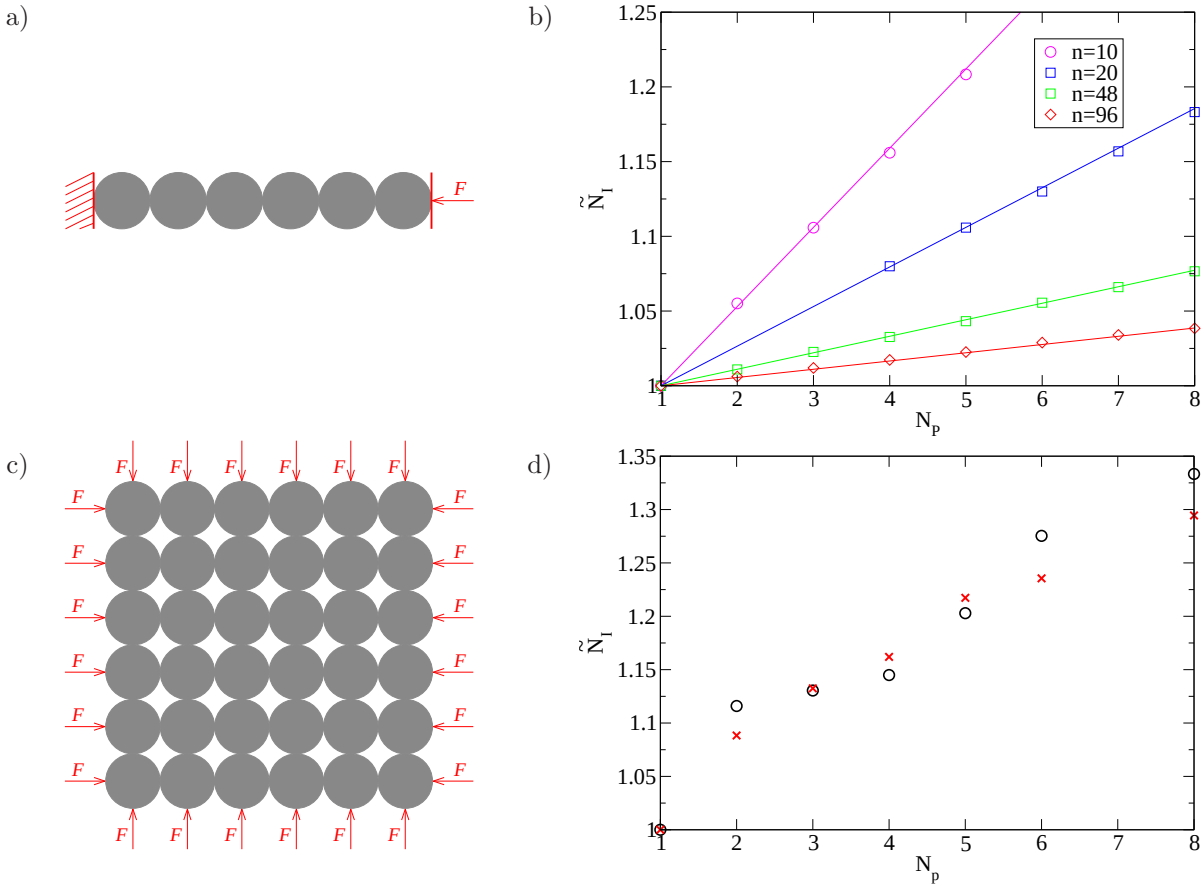

Figure 13: (color online) (a) A chain of n touching monodisperse particles, which are compressed with a force F. (b) The number of iterations needed to reach a given accuracy scaled by the value for a single processor ($\tilde{N}_I$) vs. the number of processors. The data points are simulation results, and the lines are linear fits (see text). (c) An ordered configuration of monodisperse particles, where the external forces F push the outer particles inwards. (d) $\tilde{N}_I$ vs. N_p, where open circles denote the simulation results and the crosses are the theoretical estimations.

4.3. Increase of the iteration number with the number of processors

In the iteration scheme of contact dynamics, the forces relax towards the solution in a diffusive way [36]. The diffusion constant was found to be

$$D = q\frac{4\,r^2\,N_I}{\Delta t}, \tag{16}$$

where Δt is the time step, r is the diameter of a particle, and q is a constant depending on the update method: q_p=0.5 for parallel and $q_s \simeq 0.797$ for random sequential update. Thus the diffusion coefficient of the parallel update, D_p, is smaller than that of the sequential update D_s, for a given set of parameters N_I, Δt, and r. Boundaries between sub-domains handled by different processors behave like parallel update, since the new information only arrives at the end of an iteration sweep. It is therefore expected that the same system requires more iterations in the multiprocessor version, as the number of iterations is inversely proportional to the diffusion constant.

We test this conjecture on two examples: Let us first consider a linear chain of n touching identical particles placed between two perpendicular plates [cf. Fig. 13(a)]. We suddenly switch on a compressing force on one side wall, while keeping the other wall fixed. The resulting contact forces are calculated by the iterative solver. In order to estimate the number of required iterations, we define the effective diffusion coefficient as of [44]:

$$\overline{D} = D_p p + D_s(1-p), \tag{17}$$

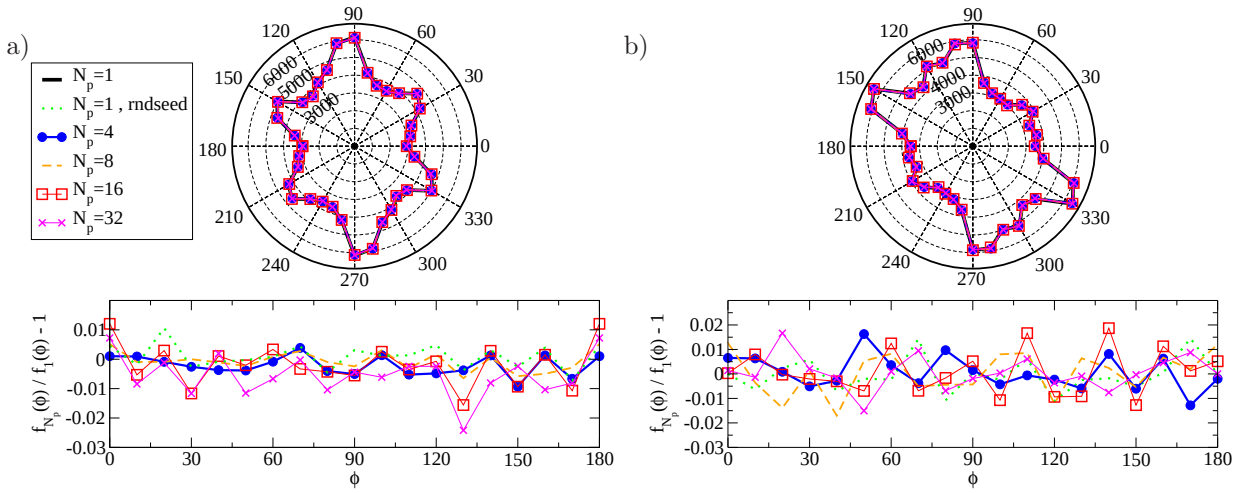

Figure 14: (color online) Angular distribution of the contact force orientations in (a) the relaxed static packing and (b) the sheared system with moving confining walls, with 8000 frictional particles calculated for different number of processors.

where p is the portion of the chain with a parallel update. In general, for each boundary one particle diameter is handled parallel and the rest sequential, which gives $p{=}N_p/n$. This is compared to the numerical results in Fig. 13(b). While in principle there is no fit parameter in Eq. (17), by adjusting the ratio to $D_s/D_p{=}1.53$ we get an almost perfect agreement for all different system sizes, as shown in Fig. 13(b). This fitted value is 4% smaller than the theoretical estimation of [36].

We have tested this scenario in a similar two-dimensional setup, where the forces were directly applied to the boundary particles as shown in Fig. 13(c). The number of iterations required for the prescribed force accuracy increases with the number of processors in a sub-linear manner [Fig. 13(d)]. This is expected as the fraction of boundary particles in a two-dimensional system scales as $\sqrt{N_p/n}$. The theoretical estimation used in the above one dimensional example with $D_s/D_p{=}1.53$ is in good agreement with the results of the two dimensional system as well. The graph of simulation results is characterized by plateaus (e.g. between $N_p{=}2{-}4$ and $6{-}8$), where the convergence rate is dominated by the higher number of domain walls in one direction.

Let us conclude here that the slower parallel diffusion part takes place in a portion $p{\propto}\sqrt{N_p/n}$ of the two dimensional system, which is negligible in reasonably large systems. For example for the medium sized system of 8000 particles, we get $p{\simeq}4\%$ for $N_p{=}16$, which would lead to about 2% increase in the iteration number. The measured value was about 1% justifying the insignificance of the iteration number increase in large systems. Indeed, we do not see a decrease in efficiency due to an increase of the iteration number for large parallel systems in Fig. 11.

4.4. Influence of the parallelization on the physical properties of the solutions

As a last check, we tested the physical properties of the system calculated by different number of processors. It is known that in the rigid limit, the force network of a given geometrical packing is not unique [45, 46]. Running the contact dynamics with different random seeds (for the random sequential update) leads to different sets of contact forces, which all ensure the dynamical equilibrium. The domain decomposition also changes the update order and the solutions will be microscopically different. Thus, a direct comparison is impossible and we have to resort to comparing distributions.

We first investigate the distribution of the contact force orientations $f(\phi)$ in the relaxed system of 8000 particles described in Sec. 4.1. The contact forces are calculated from scratch for the given geometry and boundary conditions using different number of processors. Since the system is very tall ($L_y/L_x{=}16$), it is divided only vertically for up to $N_p{=}16$, while for $N_p{=}32$ the 16 domains are cut horizontally as well. The orientation of each contact force is defined as $\phi{=}\arctan(R_y/R_x)$. The distributions of the contact force

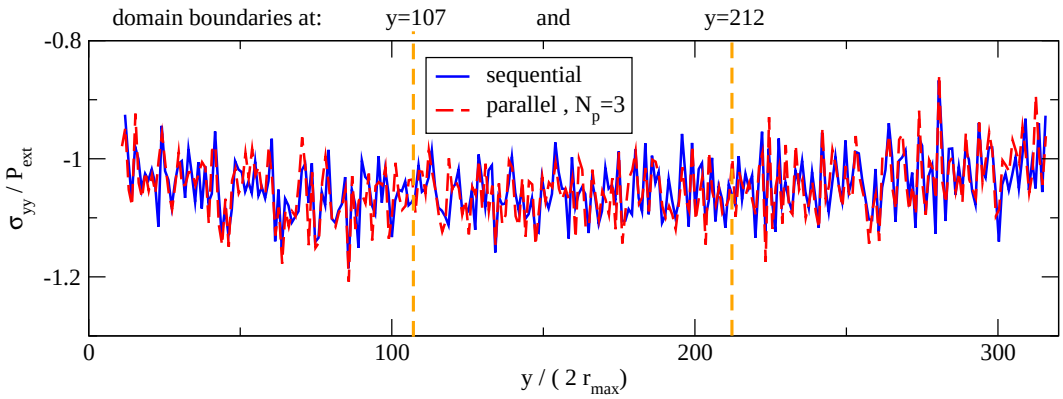

Figure 15: (color online) $\sigma_{yy}(y)$ scaled by the external pressure P_{ext} in terms of the height y scaled by the diameter of the largest particle in the system ($2\,r_{\max}$). The results obtained by the non-parallel code are compared with those obtained by the parallel code for $N_p = 3$.

orientations, $f_{N_p}(\phi)$, are compared for several values of N_p in Fig. 14(a). The range of possible values for ϕ ($[0, \pi]$) is divided into 18 bins, and each data point in the figure corresponds to total number of contacts in the same bin. For comparison, we have presented the results of the simulations with N_p=1 for two different random seeds as well. The match among the different runs are so good that the curves coincide. Hence, we also plot the relative difference $f_{N_p}(\phi)/f_1(\phi)-1$ to the non-parallel run for comparison, which shows negligible random noise. Evidently, parallelization has no systematic impact on the angular distribution of the contact forces. Similar results were obtained when the system is sheared by the horizontal confining walls moving with a constant velocity in opposite directions as shown in Fig. 14(b).

We also calculate the σ_{yy} component of the stress tensor as a function of the distance y from the bottom wall in the same system. $\sigma_{yy}(y)$ at a given height y is averaged over a horizontal stripe of width dy=$2r_{\max}$, where $r_{\max}$ is the largest particle radius in the system. The system height is thus divided into nearly 320 stripes. Figure 15 displays the results obtained by the non-parallel code as well as the parallel code with N_p=3. In the parallel case, the system is divided horizontally into three parts. The results of the parallel run match perfectly with the one of the non-parallel run. Especially, no kind of discontinuity or anomaly is observed at $y \simeq 107$ and $y \simeq 212$, where the interfaces between the processors are located.

5. Conclusion and Discussion

We have presented an efficient parallel version of a contact dynamics method in this work, which allows for large-scale granular simulations with almost 100% efficiency. We aimed at the full parallelization of the code with hierarchical domain decomposition and dynamic load balancing, in which the interface area between subdomains is also minimized. The parallel code is hence applicable to a broad range of densities and different simulation conditions.

The force calculation in CD is done by an iterative scheme, which shows an instability if more than about half of the contacts are calculated in parallel. The iteration scheme was kept domain-wise sequential while data across the domain boundaries is exchanged after each iteration sweep, ensuring that the iteration is stable for all system sizes. It is known that the CD iterative scheme approaches the solution in a diffusive manner. The diffusion constant is smaller for parallel update, which happens at domain boundaries. However, this overhead is proportional to the square root of the number of processors divided by the number of particles (in 2D), which vanishes for large systems. Regarding this as the only impact of the parallelization on the convergence, it must be expected that the efficiency is not affected by modifications at the local level i.e. non-spherical particles, three-dimensional particles, more sophisticated contact laws, etc. Of course, those can deteriorate the convergence per se but the parallel version will simply “inherit” that.

The other point of discussion raised here concerns the choice of the mesh size and adjusting the subdomain borders to it. Communication overhead was reduced because between iteration steps not all boundary

information is sent but only the relevant part of it. The subdomain wall position is only important if the particle size is not small compared to the system size. For large scale parallel applications this can only be a problem for highly polydisperse systems, for which the cell method for contact detection breaks down anyway.

The load balancing is done only at the end of each time step. Our investigations show that this happens rarely enough that load balancing overhead and CPU time fluctuations are negligible but often enough to achieve fast load balance. We used a global criterion for stopping the iteration scheme. This ensures that the physical properties of the tested samples do not show any difference compared to the non-parallel version of the code.

Blocking point-to-point communications were used to transfer data among processors. Since our algorithm needs synchronization after each iteration, non-blocking data transfer would not be advantageous. The whole amount of data is transmitted in one single packet, which reduces communication overhead over the pure data. This method introduces parallel contact update at domain boundaries, which induces an iteration number overhead due to the lower diffusivity of the information in parallel update. This overhead vanishes, e.g. with the square root of the processor number over particle number in two dimensions, which is in general negligible.

An alternative method would be to use non-blocking communications for the iteration scheme, namely to immediately send a freshly updated contact force in the vicinity of the borders to the corresponding processors, while on the other side this would trigger an interrupt when the other processor immediately updates the received contact data. This prevents the mixture of sequential and parallel update schemes. However, we do not expect that the performance of the method is greatly enhanced by the use of non-blocking communication because the information of each contact force is sent individually and the overhead associated with the increase of the inter-processor communications significantly affects the performance.

The last point to discuss concerns the load balancing method. The most exact method would be to consider the number of particles and/or contacts in each subdomain to calculate their new boundaries. Practically, this would cause difficulties, since each processor is just aware of particles and contacts within its own borders. The amount of calculations and communications between neighboring processors to place the interface according to the current contact and particle positions would make the load balancing a computationally expensive process. This lead us to balance the load further by dividing the simulation domain according to the current subdomain volumes (not always proportional to the number of particles and/or contacts), which is in fact a control loop with the inherent problems of under- and over-damping.

Acknowledgments

We would like to thank M. Magiera, M. Gruner and A. Hucht for technical support and useful discussions, and M. Vennemann for comments on the manuscript. Computation time provided by John-von-Neumann Institute of Computing (NIC) in Jülich is gratefully acknowledged. This research was supported by DFG Grant No. Wo577/8 within the priority program “Particles in Contact”.

References

[1] P. A. Cundall, O. D. L. Strack, A discrete numerical model for granular assemblies, Géotechnique 29 (1979) 47-65.
[2] S. Luding, Molecular dynamics simulations of granular materials, in: The Physics of Granular Media, Wiley-VCH, Weinheim, 2004, pp. 299-324.
[3] D. C. Rapaport, The event scheduling problem in molecular dynamic simulation, J. Comp. Phys. 34 (1980) 184-201.
[4] O. R. Walton, R. L. Braun, Viscosity, granular-temperature, and stress calculations for shearing assemblies of inelastic, frictional disks, Journal of Rheology 30 (1986) 949-980.
[5] M. Jean, J. J. Moreau, Unilaterality and dry friction in the dynamics of rigid body collections, in: Proc. of Contact Mechanics Intern. Symposium, Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1992, pp. 31-48.
[6] J. J. Moreau, Some numerical-methods in multibody dynamics - application to granular-materials, Eur. J. Mech. A-Solids 13 (1994) 93-114.
[7] M. Jean, The non-smooth contact dynamics method, Comput. Methods Appl. Mech. Engrg. 177 (1999) 235-257.

[8] L. Brendel, T. Unger, D. E. Wolf, Contact dynamics for beginners, in: The Physics of Granular Media, Wiley-VCH, Weinheim, 2004, pp. 325-343.
[9] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comp. Phys. 117 (1995) 1-19.
[10] L. Nyland, J. Prins, R. H. Yun, J. Hermans, H.-C. Kum, L. Wang, Achieving Scalable Parallel Molecular Dynamics Using Dynamic Spatial Domain Decomposition Techniques, J. Parallel Distrib. Comput. 47 (1997) 125-138.
[11] Y. Deng, R. F. Peierls, C. Rivera, An adaptive load balancing method for parallel molecular dynamics simulations, J. Comp. Phys. 161 (2000) 250-263.
[12] S. J. Plimpton, Fast Parallel Algorithms for Short-Range Molecular Dynamics, J Comp Phys, 117, (1995) 1-19.
[13] P. K. Haff, Grain flow as a fluid-mechanical phenomenon, J. Fluid Mech. 134 (1983) 401-430.
[14] S. McNamara, W. R. Young, Inelastic collapse in two dimensions, Phys. Rev. E 50 (1994) R28-R31.
[15] S. Miller, S. Luding, Event-driven molecular dynamics in parallel, J. Comp. Phys. 193, (2003) 306-316.
[16] P. Lötstedt, Mechanical Systems of Rigid Bodies Subject to Unilateral Constraints, SIAM Journal on Applied Mathematics 42 (1982) 281-296.
[17] D. E. Stewart, J. C. Trinkle, An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Coulomb Friction, International journal of numerical methods in engineering 39 (1996) 2673-2691.
[18] D-M. Mueth, H. M. Jaeger, S. R. Nagel, Force distribution in a granular medium, Phys. Rev. E 57 (1998) 3164; B. Miller, Corey O'Hern, R. P. Behringer, Stress fluctuations for continuously sheared granular materials, Phys. Rev. Lett. 77 (1996) 3110; F. Radjai, Multicontact dynamics of granular systems, Comput. Phys. Commun. 121 (1999) 294; F. Radjai, M. Jean, J. J. Moraeu, S. Roux, Force distributions in dense two-dimensional granular systems, Phys. Rev. Lett. 77 (1996) 274.
[19] D. Daudon, J. Lanier, M. Jean, A micro mechanical comparison between experimental results and numerical simulation of a biaxial test on 2D granular material, in: Powders and Grains, A. A. Balkema, Rotterdam, 1997, pp.219-222; F. Calvetti, G. Combe, J. Lanier, Experimental micromechanical analysis of a 2d granular material: relation between structure evolution and loading path, Mechanics of Cohesive-Frictional Materials 2 (1997) 121; H. Joer, J. Lanier, J. Desrues, E. Flavigny, A new shear apparatus to study the behavior of granular materials, Geotech. Test. J. 15 (1992) 129.
[20] D. Kadau, L. Brendel, G. Bartels, D. E. Wolf, M. Morgeneyer, J. Schwedes, Macroscopic and microscopic investigation on the history dependence of the mechanical behaviour of powders, Chem. Eng. Trans. 3 (2003) 979; M. Rock, M. Morgeneyer, J. Schwedes, D. Kadau, L. Brendel, D. E. Wolf, Steady state flow of cohesive and non-cohesive powders: Investigations in experiment and simulation, Granular Matter 10 (2008) 285.
[21] T. Unger, Refraction of shear zones in granular materials, Phys. Rev. Lett. 98 (2007) 018301; H. A. Knudsen, J. Bergli, Experimental demonstration of snell's law for shear zone refraction in granular materials, Phys. Rev. Lett. 103 (2009) 108301.
[22] T. M. Preclik , K. Iglberger , U. Rüde, Iterative rigid multibody dynamics, in: Multibody Dynamics 2009, ECCOMAS Thematic Conference (2009).
[23] M. Anitescu, Optimization-based simulation of nonsmooth rigid multibody dynamics, Math. Program., Ser. A 105 (2006) 113143.
[24] D. M. Kaufman, T. Edmunds, D. K. Pai, Fast Frictional Dynamics For Rigid Bodies, ACM Trans Graph 24 (2005) 946-956.
[25] K. Iglberger, U. Rüde, Massively parallel rigid body dynamics simulations, Computer Science - Research and Development 23 (2009) 159-167.
[26] K. Iglberger, U. Rüde, Massively parallel granular flow simulations with non-spherical particles, Computer Science - Research and Development 25 (2010), 105-113.
[27] A. Tasora and D. Negrut and M. Anitescu, GPU-Based Parallel Computing for the Simulation of Complex Multibody Systems with Unilateral and Bilateral Constraints: An Overview, in: Multibody Dynamics: Computational Methods and Applications, Computational Methods in Applied Sciences 23 (2011), 283-307.
[28] P. Breitkopf, M. Jean, Modélisation parallèle des matériaux granulaires, in: 4ème colloque national en calcul des structures, Giens, 1999, pp. 387-392.
[29] M. Renouf, F. Dubois, P. Alart, A parallel version of the non smooth cantact dynamics algorithm applied to the simulation of granular media, J. Comput. Appl. Math. 168 (2004) 375-382.
[30] V. Zhakhovskii, K. Nishihara, Y. Fukuda, S. Shimojo, A new dynamical domain decomposition method for parallel molecular dynamics simulation on grid, in: Annual Progress Report, Institute of Laser Engineering, Osaka University, 2004.
[31] J. K. Salmon, Parallel hierarchical N-body methods, Ph.D. Thesis, Caltech University, Pasadena, U.S.A., 1990.
[32] M. S. Warren, J. K. Salmon, A parallel hashed oct-tree N-body algorithm, in: Proceedings of Supercomputing 93, 1993, pp. 12-21.
[33] F. Fleissner, P. Eberhard, Parallel load-balanced simulation for short-range interaction particle methods with hierarchical particle grouping based on orthogonal recursive bisection, Int. J. Numer. Meth. Engng 74 (2008) 531-553.
[34] D. E. Stewart, Rigid-Body Dynamics with Friction and Impact, SIAM Review 42 (2000), 3-39.
[35] I. Nassi, B. Shneiderman, Flowchart Techniques for Structured Programming, SIGPLAN Notices 8 (1973) 12.
[36] T. Unger, L. Brendel, D. E. Wolf, J. Kertész, Elastic behavior in contact dynamics of rigid particles, Phys. Rev. E 65 (2002) 061305.
[37] D. E. Stewart, Convergence of a Time-Stepping Scheme for Rigid-Body Dynamics and Resolution of Painlevé's Problem, Arch. Rational Mech. Anal. 145 (1998) 215260.
[38] F. Jourdan, P. Alart, M. Jean, A Gauss-Seidel like algorithm to solve frictional contact problems, Computer Methods in Applied Mechanics and Engineering 155 (1998) 31-47.
[39] M. R. Shaebani, T. Unger, J. Kertész, Generation of homogeneous granular packings: Contact dynamics simulations at

constant pressure using fully periodic boundaries, Int. J. Mod. Phys. C 20 (2009) 847-867.
[40] W. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes: The Art of Scientific Computing, Chapter 19, Cambridge University Press, Cambridge, 2007.
[41] M.P. Allen, D.J. Tildeslay, Computer Simulation of Liquids, Oxford University Press, Oxford, 1987.
[42] M. Wackenhut, S. McNamara, H. Herrmann, Shearing behavior of polydisperse media, Eur. Phys. J. E 17 (2005) 237-246.
[43] V. Ogarko, S. Luding, Data structures and algorithms for contact detection in numerical simulation of discrete particle systems, in: Proc. of World Congress Particle Technology 6, Nürnberg Messe GmbH (Ed.), Nuremberg, 2010.
[44] S. Revathi, V. Balakrishnan, Effective diffusion constant for inhomogeneous diffusion, J. Phys. A: Math. Gen. 26 (1993) 5661-5673.
[45] M. R. Shaebani, T. Unger, J. Kertész, Extent of force indeterminacy in packings of frictional rigid disks, Phys. Rev. E 79 (2009) 052302.
[46] T. Unger, J. Kertész, Dietrich E. Wolf, Force indeterminacy in the jammed state of hard disks, Phys. Rev. Lett. 94 (2005) 178001.

This figure "Figure-2_600dpi.png" is available in "png" format from:

http://arxiv.org/ps/1104.3516v2


This figure "Figure-2_1200dpi.png" is available in "png" format from:
http://arxiv.org/ps/1104.3516v2